BRL
1229
c.3A

File

BRL

REPORT NO. 1229
NOVEMBER 1963

BRLESC FORTRAN

Lloyd W. Campbell
Glenn Beck

File

BRL
1229

RDT & E Project No. 1M010501A003

BALLISTIC RESEARCH LABORATORIES

ABERDEEN PROVING GROUND, MARYLAND

B A L L I S T I C   R E S E A R C H   L A B O R A T O R I E S

REPORT NO. 1229

NOVEMBER 1963

BRLESC FORTRAN

Lloyd W. Campbell
Glenn Beck

Computing Laboratory

RDT & E Project No. 1M010501A003

A B E R D E E N   P R O V I N G   G R O U N D,   M A R Y L A N D

B A L L I S T I C   R E S E A R C H   L A B O R A T O R I E S

REPORT NO. 1229

LWCampbell/GBeck/ilm
Aberdeen Proving Ground, Md.
November 1963

BRLESC FORTRAN

ABSTRACT

FORTRAN is a popular programming language that has been implemented on many computers. It is now available on Ballistic Research Laboratories' BRLESC computer. This report describes the FORTRAN language in general and includes specific details about its implementation on BRLESC.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

I. INTRODUCTION

FORTRAN is a programming language that is widely used on a variety of computers and can now be used on Ballistic Research Laboratories' BRLESC computer. FORTRAN was designed primarily for programming of scientific problems and the evaluation of arithmetic formulas. It is basically similar to the FORAST programming language that is currently used on BRLESC and ORDVAC but many of the details are different.

This manual is intended primarily for the programmers that are familiar with FORAST and BRLESC; however, it includes a general description of the FORTRAN language and should prove helpful to anyone who is interested in writing or reading FORTRAN programs. Additional details and general information can be obtained from other FORTRAN manuals and publications. The FORTRAN II manuals for the 709/7090 are suggested for those interested in using BRLESC FORTRAN since BRLESC FORTRAN is more compatible with the 709/7090 version of FORTRAN than with some versions that are used on other computers. Some readers may be surprised to learn that FORTRAN is not the same for all computers. Although the general rules are usually the same, differences in details do exist and some of the detail differences are quite subtle. It is relatively easy to write FORTRAN programs which when executed on different computers will yield different results. These differences may be due to differences in compilers or differences in the structures of the computers. However, most FORTRAN programs require only minor modifications to allow them to run on any given computer. The modifications usually require much less effort and time than would be required to re-program the problem in another programming language.

II. THE CHARACTER SET

FORTRAN allows the use of the twenty-six capital letters of the alphabet, the decimal digits 0 to 9 and the special symbols + - ( ) . * / , = . (The symbol $ is allowed only as hollerith text in a FORMAT statement.)

The card code for these characters is the same as normally used for FORAST and BRLESC except for the following:

FORTRAN	BRL	
(	%	Standard FORTRAN left parenthesis uses 0-4-8 punches that normally represents % at BRL. Since FORTRAN does not allow the 4-8 code (BRL left parenthesis) in programs, BRLESC FORTRAN allows either code to mean left parenthesis.

+	-	Standard FORTRAN uses card codes for the signs that are
-	+	just the opposite of BRL usage (+ is x, - is y at BRL). A "CHANGE + AND -" control card may be inserted in a FORTRAN program to cause BRLESC to reverse these symbols. They are initially set for BRL usage. The 709/7090 FORTRAN and BRLESC FORTRAN also allow a 4-8 card code to be a minus sign on decimal input. BRL signs are used for decimal input unless the SETMSI subroutine is used to change signs. A "CHANGE + AND -" control card does not change signs used for input data.
$	'	This card code X-3-8 is allowed only in hollerith text (H fields) in FORMAT statements.

III. SYMBOLIC NAMES AND CONSTANTS

A. General Names

In FORTRAN, all symbolic names (other than statement names) must begin with a letter and, for variables, the first letter determines the type of number it represents. Names of variables that begin with I, J, K, L, M or N represent integer numbers whereas names beginning with other letters represent floating point numbers (or Boolean variables). The length of symbolic names is restricted to six characters except subroutine names may have a terminal F as a seventh character.

B. Statement Numbers

Locations of statements (cols. 1-5 of FORTRAN statement cards) must be all decimal digits and thus look like integer numbers but are really symbolic locations of statements. (Statement numbers must be less than 32768 for 709/7090 but not BRLESC.) They do not affect the sequence of execution of the statements.

C. Constants

1. Integer constants are written without a decimal point. An integer constant on BRLESC may consist of 1 to 19 decimal degits (less than 2^{64} in absolute value) except for those integers involved in division operations which must be less than 2^{34} in absolute value. Some computers restrict integer constants to as few as four decimal digits. The values of integer variables are restricted to the same maximum value as integer constants on BRLESC and most computers.

2. Floating point constants must be written with a decimal point. They may consist of a decimal point with 1 to 19 decimal digits (on BRLESC) and may be followed by an E and a decimal exponent. The BRLESC range of floating point

constants (variables) is between 10^{155} and 10^{-155} approximately in absolute value with zero also allowed. Most computers have a more restricted range of numbers.

Examples: 1. , 4.21 , .2 , 51.6 E2 , .1E-3

3. Boolean constants are written as twelve octal digits. If less than twelve digits are written, zeros are added by the computer to the left of the digits to make a total of twelve.

D. Arrays

Blocks of storage are referred to as arrays in FORTRAN and are defined in DIMENSION statements. Subscripts are enclosed in parentheses in FORTRAN, e.g. A(3) or B(I,J), and one, two, or three dimensional arrays may be used. Any subscript may be variable and "indexing", as done in FORAST, is not allowed. Subscription of variables is done by substitution rather than addition and the lower bound of all subscripts is one. Subscript arithmetic is allowed; BRLESC FORTRAN allows any integer arithmetic expression that does not itself involve any subscripted variables, however, the most general expression allowed in standard FORTRAN is C * V ± C' where C and C' are integer constants and V is an integer variable.

Subscripted names must not end with F unless they have three or less characters.

Symmetric arrays are not allowed in FORTRAN and there is no provision for "interweaving" arrays.

E. Absolute Addresses

There is no provision in FORTRAN for writing absolute addresses.

IV. ARITHMETIC EXPRESSIONS

The following symbols denote the following operations:

+	addition
-	subtraction
*	multiplication
/	division
**	exponentiation

The use of functions (subroutines with only one result) is also allowed by writing the name of the function in front of parentheses that enclose the arguments. (FORTRAN allows functions to have more than one argument and commas are used to separate the arguments.) The arguments may be arithmetic expressions.

The precedence of operations when not governed by the use of parentheses is

functions (subroutines)
**
* and /
+ and -

where the operations higher or the list will be performed before those that are lower on the list. For successive + and - operations or successive * and / operations, they will be performed from the left to the right. Parentheses may always be used to cause the operations to be done in any desired sequence. Successive exponentiations must always have parentheses to show the desired grouping.

The major difference between FORAST and FORTRAN arithmetic expressions is the grouping of successive multiplications and divisions. FORAST groups them from the right and FORTRAN groups them from the left. Thus in the expression (A * B/C * D), D is in the denominator for FORAST and is part of the numerator for FORTRAN.

Implied multiplication should not be used in FORTRAN (although some versions do allow it and BRLESC FORTRAN allows it after a right parenthesis).

<u>Fixed Point Fractional Arithmetic Is Not Allowed in FORTRAN.</u> All arithmetic within an expression must be one mode (integer or fl. pt.) except for integer subscripts and integer powers of exponentiation in floating point expressions. The first letter of the names and the use of the decimal point in numbers determines the mode rather than preceding the expression with a declaration of the mode as is done in FORAST.

Parentheses <u>must</u> <u>not</u> be omitted at the ends of an expression. The number of left parentheses must be the same as the number of right parentheses in each expression.

Two operations must not appear adjacent to each other in formulas; e.g. / - or * * - .

Any operation on integers which does not yield an exact integer result is truncated except negative integer results of division on BRLESC FORTRAN will give the greatest integer that does not exceed the algebraic exact result. Thus -4.2 will give -5. This is probably different than results on the 7090 or other absolute value machines.

For 709/7090 and BRLESC FORTRAN, boolean expressions are allowed and are designated on cards with a B in column one. The symbols +, *, - denote the logical operations of <u>or</u> (inclusive), <u>and</u>, and <u>complement</u> respectively. BRLESC FORTRAN performs these operations only on the rightmost 36 bits of a word so that it is compatible with the 36 bit word length of the 709/7090. The leading 32 bits of a BRLESC word will be zeros after a logical operation.

Double Precision arithmetic expressions are allowed (a D in col. 1) but are done in BRLESC single precision which is as accurate as 709/7090 double precision.

Complex arithmetic expressions are <u>not</u> presently allowed in BRLESC FORTRAN. An I in column one will cause an error print.

V. ARITHMETIC FORMULAS

The general form of FORTRAN arithmetic formulas (arithmetic statements) is

v = ae

where v is a name of a variable (it may be subscripted) and ae is an arithmetic expression. An example would be

```
X(J + 1) = A(J)**2 - V/(T + 3.)
```

The arithmetic expression is evaluated and the result is stored as the new value of the variable whose name is on the left of the = symbol.

No arithmetic may be performed on the left of the = symbol except for subscript arithmetic. Only one = symbol is allowed and hence only one variable will have its value changed by an arithmetic formula.

The arithmetic expression may be just a name of a variable or constant, e.g. X = A.

VI. SPECIFICATION STATEMENTS

This group of statements (DIMENSION, EQUIVALENCE, COMMON and FREQUENCY) provides information to the compiler and may be used by the programmer to control the storage assignment of some or all of the variables. These statements do not cause any machine code to be generated for running the program, they only affect the way it is compiled.

1. DIMENSION a(i), b(i1,i2), c(i3,i4,i5), ..., where a,b,c are array names and the i's are integer constants.

This statement is used to declare the names and maximum sizes of arrays. The <u>maximum</u> subscripts are enclosed in parentheses and they must be decimal integer constants, not variables. The minimum subscript is always taken to be <u>one</u>. One, two, or three dimensional arrays may be defined in any sequence.

Example: DIMENSION T(41),X(10),E(4,4,4),A(3,7)

2. EQUIVALENCE (a,b,c,....),(d,e,f,.....), where a,b,c,d,e,f are names of any type of variable.

This statement causes different names to be assigned to the same memory space. (It performs the same function as SYN does in FORAST.) All the names within a set of parentheses are made equivalent. Increments may be used if desired by enclosing them in parentheses immediately after the name. (No increment is the same as an increment of one.)

3. COMMON a,b,c,d,e ...., where a,b,c,d,e are the names of variables of any type.

This statement allows the programmer to specify that certain variables and arrays are the same in more than one program or subprogram (subroutine or function). The storage assigned to those items in the COMMON statement in one subprogram is the same storage assigned to the items in the COMMON statements in all of the other subprograms (and also the main program). Thus it also has an equivalence effect between subprograms. All storage used in each subprogram is different than the storage in any other subprogram except for the items that are listed in COMMON statements.

Within each subprogram, all COMMON variables are assigned consecutively in the sequence in which they appear. The starting point for all the subprograms within each total program is the same. Proper space is left for arrays.

COMMON statements are used to avoid listing many arguments when using a subprogram. By forcing subprograms to use the same storage as the main program for some (and possibly all) of the variables, the need for specifying and moving variables is removed.

If any COMMON variable also appears in an EQUIVALENCE statement, the <u>COMMON assigning has priority</u> and is done first in BRLESC FORTRAN. <u>This is different than 709/7090 FORTRAN</u> where EQUIVALENCE variables are assigned first and will change the sequence of storage assigned to COMMON variables. BRLESC FORTRAN handles the COMMON-EQUIVALENCE interaction as specified for FORTRAN IV and in fact allows labeled (block) COMMON and array definitions in COMMON the same as FORTRAN IV.

4. FREQUENCY.

This statement is ignored by BRLESC FORTRAN. Its purpose in 709/7090 FORTRAN is to provide information that helps the compiler to optimize the program.

VII. CONTROL STATEMENTS

This group of statements provides for controlling the sequence in which statements are executed in the running program. Unconditional transfer of control (sometimes called branching or jumping) is provided for by several types of GOTO statements and conditional transfer of control is provided by several types of IF statements. A DO statement allows definition of a "loop" and a CALL statement causes transfer of control to a subroutine with a return to the next statement. There are two statements (STOP and PAUSE) that cause the program to stop running.

1. GOTO s where s is a statement number.

This statement causes the statement numbered s to be done next.

Example: GOTO 22

2. GOTO (s1,s2,s3,...),i where s1,s2,s3 are statement numbers and i is a nonsubscripted integer variable.

This is referred to as a "computed GOTO" and the statement done next depends on the value of i. If i = 1, s1 is done next; if i = 2, s2 is done next; etc.

Example: GOTO(4,19,462),K

3. ASSIGN s TO i where s is a statement number and i is a nonsubscripted integer variable.

This statement causes the address of the statement numbered s to be put into the integer variable i and this type of statement is to be executed before the "assigned GOTO" statement (as explained in the next paragraph) is executed.

Example: ASSIGN 64 to M

4. GOTO i, (s1,s2,s3,....) where i is a nonsubscripted integer variable and s1,s2,s3, are statement numbers.

This statement transfers control to the statement that has the number that was last assigned to i by means of an ASSIGN statement. The (s1,s2,s3....) enumeration in this statement is not really necessary but should be used to list the possible statement numbers that this "assigned GOTO" statement may transfer control to.

Examples: ASSIGN 44 to N
GOTO N, (16,29,44,192)

5. IF (ae) s1,s2,s3 where ae is an arithmetic expression and s1,s2,s3 are statement numbers.

This statement causes control to be transferred to statement s1,s2, or s3 depending on whether the value of the arithmetic expression ae is negative, zero (exactly), or positive respectively.

```
Examples:  IF(X)4,7,22
           IF(R * V - 4.1*(U+V))16,244,16
```

6. DO s i = i1, i2, i3 where s is a statement number, i is a nonsubscripted integer variable and i1,i2,i3 are integer constants or nonsubscripted integer variables.

This statement causes the statements following this DO statement up to and including the statement numbered s, to be executed repeatedly with the integer variable i initially assuming the value of i1. The variable i is incremented by i3 at the end of the sequence of statements and the sequence is repeated if the value of i does not exceed i2. The specification of i3 is optional. If i3 is not specified, its value is taken as one (1).

A DO sequence of statements may itself contain a DO sequence provided the entire inner DO sequence is contained in the outer one. Several DO's may terminate on the same statement.

While most versions of FORTRAN have several other restrictions concerning DO loops, BRLESC FORTRAN does not have any other restrictions on the construction of the statements that are included in DO loops. BRLESC FORTRAN does always set and use the actual integer variable specified for i in a DO statement and its final value (plus the increment for normal termination of the DO loop) is always stored there.

```
Examples:  DO 42 K = 1,L
           DO 3 JT = MIN, 55, NSTEP
```

7. CONTINUE

This is a dummy statement that generates no object code except when it is the last statement in a DO loop. Its statement number is always used if it has one. It must be used as the last statement in a DO loop whenever the last statement would have been an IF or GOTO type of statement that transfers control. Whenever a CONTINUE statement is the last statement in a DO loop, its statement number is the location of the machine instructions that increment the DO variable and test it for its maximum value.

8. STOP or STOP w where w is an octal constant that is ignored.

This statement causes the running program to be terminated and should only be used to indicate that the program has run to completion. This statement causes BRLESC FORTRAN to empty the tape output buffer, rewind the standard output tape if it should be rewound, check for overflows and halt at N40. On BRLESC, the program may also be terminated by reading a card or tape line that has the first ten characters of either "ENDbTAPEbb" or "bbbbbbPROB" where b represents a blank.

Examples: STOP
STOP 77

9. PAUSE or PAUSE w where w is an octal constant.

This statement causes the program to halt and display the octal constant. (BRLESC displays it in the α address of the halt order.) If the computer is re-started manually by pressing the proper button (initiate on BRLESC), the program will continue with the next statement.

This statement should not be used without a very good reason for using it.

Examples: PAUSE
PAUSE 421

10. CALL a(b,c,d,....).

This statement causes the subroutine named "a" to be entered and executed with b,c,d... as the arguments, parameters, and store addresses. (Arithmetic expressions are allowed.) For BRLESC FORTRAN, "a" could be the name of a function (a subroutine with one result) and the subroutine being called must be one that is a standard one on the compiler tape or one whose code is included in the program as a SUBROUTINE (or FUNCTION).

The arguments used in a CALL statement must agree in mode with the mode of the dummy variables that were used when the subroutine was defined. If there are no arguments, they may be omitted.

CALL EXIT or CALL DUMP statements on BRLESC are the same as a STOP statement. CALL CHAIN causes an error print and CALL PDUMP is ignored.

Alphanumeric arguments are not allowed in BRLESC FORTRAN.

Examples: CALL SUB3(X,Y,R)
CALL TOTAL

11. IF (SENSE SWITCH i) s1,s2.

This statement transfer control to statement s1 or s2 if sense switch i ($1 \leq i \leq 6$) is down or up respectively. (i must be a constant.) On BRLESC, the manual read switches 15-20 are used as sense switches 1-6 respectively. However, these switches may be "preset" by a program control card to be either "down" or "up" regardless of their actual position. (See SETSSW in Section XV.)

Example: IF (SENSE SWITCH 3)14,92

12. SENSE LIGHT i where i is 0,1,2,3, or 4.

If i is 0, then all sense lights are turned off. If $1 \leq i \leq 4$ (actually 6 on BRLESC), sense light i only will be turned on. The rightmost four (actually six) bits of cell 062 on BRLESC are used as sense lights. Initially on BRLESC, all of them are off.

Example: SENSE LIGHT 2

13. IF(SENSE LIGHT i)s1,s2

If sense light i is on, statement s1 is done next otherwise statement s2 is done next.

Example: IF (SENSE LIGHT 2)67,39

14. IF ACCUMULATOR OVERFLOW s1,s2

This statement checks for fixed point add-subtract or shift overflow on BRLESC and does statement s1 next if it has occurred. Otherwise statement s2 is done next. (The very last operation may not be included in the check on BRLESC and this test turns the indicators off if they were on before.)

15. IF QUOTIENT OVERFLOW s1, s2

This does exactly the same as the IF ACCUMULATOR OVERFLOW statement explained above.

16. IF DIVIDE CHECK s1,s2

On BRLESC FORTRAN, this statement checks for floating point division by zero (or unnormalized divisor) or fixed point division overflow. If either has occurred in the program, statement s1 is done next; otherwise s2 is done next. (The very last operation in the previous statement may not be included in this test on BRLESC and this test turns the indicators off if they were on before.)

VIII. THE FORMAT STATEMENT

FORMAT (Special Specifications)

This statement is not executed but is used to specify the field lengths, spacing and the form of the data for either the reading of input data or the printing (or punching) of output data. It is always used in conjunction with one of the input-output statements and does nothing by itself.

Let n = number of times to repeat this field. (n is optional, used as 1 if not specified.)

w = the width of the field (the number of columns or characters).

d = the number of decimal places to the right of the decimal point. (d is used modulo 10 on 709/7090 but not on BRLESC.)

Then the types of fields that may be specified are:

nIw	for integer numbers.
nEw.d	for floating point numbers with exponents.
nFw.d	for floating point numbers without exponents.
wX	for spacing or blank columns.
nAw	for alphanumeric fields.
wH	for alphanumeric (hollerith) fields where the characters are read into or printed from the w characters following the H in the FORMAT statement itself.
nOw	for Octal numbers.

Consecutive field specifications are separated by commas, thus "FORMAT (I6,3E14.6, F10.7)" is an example of a FORMAT statement. Each complete FORMAT statement specifies the maximum length of the record (card or printer line) that will be read, printed or punched when that FORMAT is used.

Two sets of parentheses are allowed in 709/7090 FORTRAN and four sets are allowed in BRLESC FORTRAN so that groups of specifications may be repeated within a FORMAT statement. A left parenthesis may be preceded by an integer n to indicate the number of times to repeat the specifications enclosed in parentheses. Thus FORMAT (E12.5,3(I6,F9.3)) would be a format where the I6,F9.3 portion would be repeated three times.

If the input-output statement list contains more items than specified by the FORMAT being used, then a new card or line is begun and the FORMAT is repeated from its rightmost left parenthesis. If this parenthesis is preceded by a repeat number, it will be used in BRLESC FORTRAN while the 709/7090 manuals do not indicate that it would be. If the FORMAT specifies more fields than required for an input-output list, the rest of the FORMAT is ignored except any H fields that follow the last number will be used.

A slash "/" may be used in a FORMAT statement to indicate that a new card or line should be started. Thus FORMAT (I10/E15.6) used for punching cards would cause a ten column integer to be on one card and a fifteen column floating point number to be on the next card. If a slash is used where a new line starts anyway, it is ignored except N+1 consecutive slashes will always cause N blank lines or cards (or skip N cards for input).

Scale factors may be used with F type specifications (and in a limited way with E type specifications). An integer, s, specifies the power of ten (scale factor) to multiply the internal number by to obtain the external number, i.e. input numbers get divided by 10^s (not on BRLESC) and output numbers get multiplied by 10^s. The integer s is written in front of the nFw.d specifications and the letter P is used to separate s and n, e.g. -2P4F10.5 or -2PF15.5 specify a scale factor of 10^{-2}. On BRLESC FORTRAN, either a + or a - sign in front of s is used as a minus sign. <ins>Therefore never write + signs in front of s.</ins> Once s has been specified, the scale factor remains in effect for the rest of that FORMAT statement (including repetitions) and will be used on subsequent E and F type fields. A OP specification may be used to reset it to 0. For input, a punched decimal point overides both the scale factor and the d specified. For E fields, only a positive scale factor may be used and it does not change the value of the number, it only indicates that s digits should be printed in front of the decimal point. (It has no meaning for input E fields.) Thus the number 2 would normally print 0.20E01 for s = 0, but for s = 1, it would print 2.00E00 and s = 2 would print 20.00E-01.

<ins>I Fields</ins>

<ins>Input</ins>: Most FORTRAN compilers assume the integer to be punched at the <ins>right</ins> end of the field without a decimal point; however, BRLESC FORTRAN will accept it any place within the field and it may have a decimal point.

<ins>Output</ins>: The integer will be punched at the right end of the field with a floating sign. (All output has a floating sign which means that the sign is in the column preceding the leftmost digit that is printed. Leading zeros are not printed on I or F fields.)

<ins>E Fields</ins>

<ins>Input</ins>: The number may or may not have an exponent. An E or a sign, but not a blank, may be used to indicate the starting of the exponent. The exponent may be less than four columns. If a decimal point is punched, it is used and overides the s and d specification. If no decimal point is punched, then it is assumed to be after d digits (columns)

left from the start of the exponent. Most FORTRAN compilers require that the number be punched at the right end of the field, but BRLESC FORTRAN allows it anywhere within the field. Blank columns are used as zeros (except after the exponent on BRLESC).

Output: The floating point number will be printed with a four column exponent that includes an E, a sign, and two digits for the value of the exponent. A decimal point is printed d digits from the right end of the coefficient and if s = 0, a zero is printed in front of the decimal point. If $s \geq 1$, then s digits of the coefficient are printed to the left of the point. The sign immediately precedes the first digit printed. The entire number is printed at the right end of the field of w columns.

F Fields

Input: The same as E fields, see above. (This may not be strictly true for 709/7090 but will generally give the desired result except possibly for the use of a scale factor.)

Output: The floating point number will be printed without an exponent and the decimal point will be printed d digits from the right end of the field. The actual number printed is 10^s times the number that is in the computer.

H Fields

Input: The alphanumeric information is stored in the FORMAT statement itself immediately following the H. No transformation of characters is done, the sign option setting for numeric input on BRLESC has no effect on H fields.

Output: The w alphanumeric characters that immediately follow the H are printed. Blanks are not ignored and there is no transformation of any of these characters. Thus on BRLESC, "(+ - " characters will not be the ones intended if the deck was punched using standard FORTRAN characters at some other installation. (The "CHANGE + AND -" control card does not change the + and - signs in H fields.) For tape output, if an H field occurs at the beginning of a line, the first character is used for vertical high speed printer format control instead of actually getting printed.

A Fields

Input: This causes w alphanumeric characters to be stored in the variable name that is on the input list. To be compatible with 709/7090, BRLESC FORTRAN stores a maximum of six characters per word at the right end of the word. If $w < 6$, the characters will be at the left of the 36 bits with blanks to fill out the word. For BRLESC only, if $w > 6$, then the additional characters will be stored in the next consecutive word(s), six per word. As with H fields, no transformation of characters is done. This can be used to read FORMAT specifications at run time.

Output: This causes w alphanumeric characters to be printed from the contents of the variable name that is on the output list. The rules listed above for A input are followed so that whatever is read will be printed exactly the same.

X Fields

Input: This causes w columns to be skipped whether they are blank or not.

Output: Causes w blank columns to be printed.

O Fields

Input: This allows octal numbers to be read and stored at the right end of BRLESC words in the same manner as integers. (There is no left normalization.)

Output: This allows integers (octal or decimal) to be printed in octal form at the right end of the field.

FORMAT statements may be placed anywhere within a program (or subprogram) except as the first statement within a DO loop. (This restriction does not apply to BRLESC but should be followed. On BRLESC, FORMAT statements are done as NOP instructions so it is best not to place them where they will be done often.) FORMAT statements are kept as alphanumeric information and decoded at run time, thus it is permissible to use A fields to read FORMAT statements (without the word FORMAT) at run time. The variable names of such statements must be listed in a DIMENSION statement for 709/7090 but is not required for BRLESC.

If the list in an output list is exhausted and the next item in a FORMAT statement is an H field, the H field is printed. (If the end of FORMAT and list occur at the same time and an H field follows the last left parenthesis, it will not be printed.) Note that a FORMAT may contain nothing but one or more H fields.

Blank characters in a FORMAT statement are ignored except within H fields. The w count for an H field must include the blanks within the H field.

The comma separating field specifications may be omitted when it follows an H or X field specification or would precede or follow a parenthesis or slash. (This rule may not hold for all computers but is true for BRLESC.)

```
          Examples:  FORMAT(3I5,(E15.8))
                     FORMAT(2HX=,F10.4,4(1PE12.5))
                     FORMAT(6F10.4/4I10//)
```

IX. DESCRIPTION OF INPUT-OUTPUT LISTS

The names of the variables to be transmitted between the computer and the input-out devices are specified on a list in the proper type of input-output statement and the sequence of the names on the list determines the sequence of transmission. Simple variable names, subscripted array names where the subscript control is either specified in other statements or within the input-output list, and array names without subscripts are allowed. Array names without subscripts cause the entire array to be transmitted and the elements must (for input) or will (for output) be arranged in the same sequence that they have in the computer memory. (BRLESC and 709/7090 vary the subscripts from left to right, thus two dimensional arrays are stored by <u>columns</u>; i.e. A(1,1), A(2,1), A(3,1) etc. is the sequence of elements of the array A.) Commas are used to separate the names on an input-output list.

Indexing information specified within the list is written <u>after</u> the names of variables to which it applies and the names and the indexing information are all enclosed in parentheses. For example A, (B(I), I = 1,10) would cause the transmission of A, B(1), B(2), ... B(10). Note that the indexing information is written the same as in a DO statement with the increment taken as one if it is not written. It is permissible to nest these parentheses, e.g. ((A(I,J),I = 1,5), J = 1,5). Note that commas are used to separate items on the list and must be used after a right parenthesis except for the last one. The indexing within each set of parentheses is done to completion before going on to the next indexing specification.

All of the input-output statements that transfer alphanumeric (not binary) data make use of a FORMAT statement to specify the field types and lengths. The type (integer or floating point) of a name specified on an input-output list must correspond to the type of field specified in the FORMAT statement that is being used. All integer variables must use I fields and all floating point variables must use E or F fields. (BRLESC does allow integers to be printed as integers in E or F fields.) The FORMAT controls the maximum length of each line. A line is shorter than specified in a FORMAT only when the end of the list is reached before the end of the FORMAT. Whenever the end of the FORMAT is reached before the end of the list, the FORMAT is repeated from the last right parenthesis and a new line (or card) is started. (See Section VIII for more information about FORMAT statements.)

Constants and arithmetic expressions are not permitted on input-output lists except indexing information may contain constants and subscripts may be constant or arithmetic expressions.

It is permissible to read an integer variable and use it as a subscript within the same input list if its name is separated from the place it is used by at least two left parentheses. (This is counting the one used to indicate a subscripted variable. Extra parentheses may be used just to meet this requirement.) Thus J,(B(J)) is an example where the value of the variable J just read will be used as the subscript for B(J). (For BRLESC, the extra parentheses are not required if two or more variables or any indexing information separates the integer from where it is used.)

```
Examples:  A, B, I
           N, M, (BA(N)),P
           ((A(I,J), J = 1,10), I = 1,10), (R(K), K = 2,20,2)
```

X. INPUT-OUTPUT STATEMENTS

The following group of statements may be used in FORTRAN to control the flow of information between the computer and input-output devices or secondary storage. Card reading or punching, magnetic tapes and, on some computers but not on BRLESC, drums may be used to read or record data. Most of the statements also use a FORMAT statement to control the conversion of data between computer form and printer or card form. However, the READ TAPE and WRITE TAPE (and the corresponding DRUM statements on computers that allow them) cause the transfer of data without any conversion. This computer form of data will be referred to as binary information and actually is binary numbers for a binary computer such as BRLESC. The other statements cause the reading or printing of data in alphanumerical form. There are three statements, END FILE, REWIND and BACKSPACE that do not transfer data but can be used to manipulate the magnetic tapes.

In all of the input-output statements described below:

f	is a FORMAT statement number or name.
"list"	is any allowable input-output list (See Section IX).
t	is a magnetic tape number or variable. (See BRLESC restrictions on t at end of this section.)

READ f, list

This statement causes decimal and alphanumeric data to be read from cards (or tape 6 on BRLESC if the cards have been put on tape off-line and console switch 36 is up.) (BRLESC may use all 80 columns for either input or output cards.) If the list is omitted on this statement, one card will be read and ignored on BRLESC.

PUNCH f, list

This statement causes decimal and alphanumeric data to be punched on cards (or actual tape 13 if console switch 38 is up. The tape output will be "formatted" for the high speed printer by adding a 1 character at the beginning of each "card" and an end-of-line character "Δ" at the end of each "card". The block length will be at least 860 characters.) All 80 columns of a card may be used on BRLESC and for tape 13 output, the "card" may be up to 160 columns long.

PRINT f, list

For most computers, this statement means to print the data on an on-line printer. Since BRLESC does not have an on-line printer, the data is put on actual tape 13 for off-line printing. The maximum line length for most computers is 120 characters, but is 160 characters on BRLESC.

The following description generally applies only for BRLESC. If the first character of a line comes from an H field, it will be used for vertical format control (after a transformation) and not printed. If the first character does not come from an H field, an extra "1" character (single space) is inserted at the beginning of the line. In either case, the zone bits will be set to 01 so that it is possible to print PRINT output separately from PUNCH output when both are on tape 13. The end-of-line character "Δ" is automatically inserted at the end of each line. The tape writing is parity checked but there is no checking for end of reel. The tape block length is at least 860 characters and this allows about 5 million characters or 60,000 lines of 80 characters each on a reel of BRLESC tape.

For BRLESC, a control card may be used to change all PRINT statements to PUNCH statements.

READ INPUT TAPE t, f, list

This statement causes decimal and alphanumeric input data to be read from tape t. Each "line" of data is assumed to be a separate record (block) on tape and may be as long as 120 characters (160 on BRLESC). If the tape was previous output that has a vertical format character at the beginning of each line, provision must be made in the FORMAT for skipping that character. (If the tape was previous BRLESC output, the first character is probably not a blank.) The tape reading is parity checked on BRLESC but there isn't any checking for end of reel.

If the "list" is omitted with this statement, it will cause one line to be read and ignored on BRLESC.

Just INPUT may be used instead of READ INPUT TAPE in this statement on BRLESC.

WRITE OUTPUT TAPE t, f, list

This statement causes decimal and alphanumeric output data to be recorded on tape t. Each line of data is recorded as one record (block) on the tape and may not exceed a total of 120 characters (160 on BRLESC). The first character will be used as a vertical format control for the high-speed printer and is determined in the same manner as for the PRINT statement described above except the zone bits will be 00.

The tape writing is parity checked on BRLESC but there is no checking for the end of a reel. About 9500 lines of print will fill a BRLESC tape.

Just OUTPUT may be used instead of WRITE OUTPUT TAPE in this statement on BRLESC.

READ TAPE t, list

This statement causes <u>binary</u> information to be read from tape unit t. It should be used only for reading data that was previously put on tape by the use of the WRITE TAPE statement described below. This statement will not read more data than was specified on the list of the statement that wrote the data. (Such a group of data is defined to be a "logical record".) If less than the entire logical record is read, the tape will move to the end of the record. (If the list is omitted entirely, the tape still moves to the next logical record.) If an attempt is made to read more data than is on one logical record, the unused portion of the list will be ignored.

On BRLESC, binary logical records are subdivided into tape blocks of 128 words each. Within each logical record, the first word of each tape block is zero except for the last block. The first word of the last block contains in the α address the number of words in the last block (not counting this word) and in the γ address, the total number of blocks in the logical record.

WRITE TAPE t, list

This statement causes all of the data specified on the list to be written as binary information in one logical record (see READ TAPE) on tape unit t. It is useful for temporarily recording data on tape that may be read back into the computer by using a READ TAPE statement at a later time. See the explanation of the READ TAPE statement for a description of the way the information is "blocked" on the tape.

END FILE t

This statement causes a file mark to be written on tape t.

BACKSPACE t

This statement causes tape t to be moved backward one "logical record". This is all of the data written by the WRITE TAPE statement that wrote the record for a binary tape, or is one line (or "card") if it is an INPUT or OUTPUT tape.

REWIND t

This statement causes tape t to be rewound without being interlocked.

READ DRUM i, j, list

This statement is not allowed on BRLESC and causes an error print. For the 709/7090, it means to read data from drum i beginning at the jth word. (Variable subscripts are not allowed on the list.)

WRITE DRUM i, j, list

This statement is not allowed on BRLESC and causes an error print. For the 709/7090, it means to write data on drum i beginning at the jth word. (Variable subscripts are not allowed on the list.)

Additional Notes on Input-output Statements:

The f (FORMAT number or name) may be omitted in READ, PUNCH or PRINT statements on BRLESC and this will cause a FORMAT (1P6E12.5) to be used automatically.

The statement numbers 1 and 2 may be used to automatically specify FORMAT (5F14.5) and FORMAT (1P5E14.5) respectively without including them as part of the program. If 1 or 2 or both are used to refer to these FORMATS, then that statement number must not be used in the program for any other purpose. If either one is used as a statement number in a program, then the corresponding automatic FORMAT cannot be used.

The omission of a "list" on any of the input statements will cause one record (card, line, or logical binary tape record) to be read and ignored on BRLESC. Some computers may allow the FORMAT specified to be used to skip more than one record. Note that a FORMAT should be specified when the list is omitted although it is not necessary to do so on BRLESC.

The number of print positions on BRL's high-speed printer is 160. BRLESC FORTRAN allows at most a total of 170 characters for a line including the vertical format and end-of-line characters. When the 1401 can be used to print BRLESC tapes, the line length will probably be restricted to 132 characters.

ADDITIONAL NOTES ON THE USAGE OF MAGNETIC TAPE ON BRLESC:

All of the tape reading and writing is parity checked. Rereading erroneously ten consecutive times or rewriting wrong twice after each of five consecutive "GAP instructions" causes an error print and BRLESC stops running the programs.

There is no checking for end-of-reel conditions in BRLESC FORTRAN. Single line per block INPUT or OUTPUT tapes will only hold about 9500 lines while PRINT or PUNCH tape output will hold about 60,000 lines per reel.

The only restriction on switching between reading and writing of tapes is that writing should only begin at the beginning of a tape. (If writing is started in the middle of a tape, an extra "block" of random bits may be written on the tape.) Whenever a tape on BRLESC is switched from writing to reading, a file mark and an extra one word block that says "END TAPE" is automatically written on the tape before the final file mark is written and then switching is done. (This extra block is ignored by a BACKSPACE statement.)

The tape unit number t may be either a decimal integer constant or variable. If t is a variable, the integer value it has at the time the tape statement is executed is used as t. The following table shows the correspondence between the value of t and the tape switch on BRLESC. The actual tape handler used depends on the switch setting.

t	Switch
1 or 11	1
2 or 12	2
3 or 13	3
4 or 14	4
5 or 15	5
6	9
7	10
8	11
9	12
10	7 (temporary or output only)

Note that $1 \leq t \leq 15$ and that t is used modulo ten for $11 \leq t \leq 15$. (If $t > 15$ is used, it will be used modulo 16.) PRINT (and PUNCH) tape output uses switch 13, the compiler itself uses switch 15 for its own program and uses switch 7 for temporary storage while compiling, and card input put on tape off-line uses switch 6. (Usage of switches 6,7 and 15 are identical to FORAST.) When leaving problems to be run on BRLESC, the switch number rather than the t number should be used in the instructions to the computer operator.

All printer output is formatted for variable length lines for the off-line high speed printer. PUNCH tape output automatically has a single space character "1" inserted at the beginning of each line and an end-of-line character "Δ" at the end of each line. The same is true of PRINT and WRITE OUTPUT TAPE output if the first field of the line is not an H type field. If the first field is an H field, then the first character of the field is used for vertical format control after undergoing the following transformation:

H Field	Tape
+ or -	blank (no space)
0 or 2	2 (double space)
1 or 8	8 (start new page)
Others	1 (single space)

The zone bits of the format character will be 00 except PRINT output will have 01.

FORTRAN programs are supposed to contain an END FILE statement and a REWIND statement for each output tape used in the program and a REWIND statement for each input tape. If this is not done at the end of the program, the operator will have to manually do these operations and the output tape will not have a file mark at its end. PRINT (or PUNCH) tape output on BRLESC is automatically completed and rewound unless PAUSE is used to halt the problem instead of STOP, CALL EXIT, or CALL DUMP.

XI. SUBPROGRAM STATEMENTS

FORTRAN allows sections of a FORTRAN program to be designated as subroutines that may be used at many different places in the main program. The SUBROUTINE, FUNCTION, RETURN and END statements allow the programmer to define and name portions of his program as subprograms and they provide information that allows the compiler to provide for the substitution of variables at run time and standard entry and exit methods used for subroutines.

Any subprogram may use any of the FORTRAN statements within itself except SUBROUTINE and FUNCTION statements. Any subprogram may use any other subprogram or subroutine of any type, including arithmetic statement functions (see Section XII) that are defined at the beginning of the subprogram. Recursive subprograms (subprograms that use themselves) are not allowed.

1. SUBROUTINE a(b,c,d,e,....)

This statement marks the beginning of a subprogram that we shall call a SUBROUTINE. The name of the SUBROUTINE is a and b,c,d,e,... are the names of nonsubscripted dummy variables that will be replaced at run time by the actual variables that are listed in the CALL statement that causes this subroutine to be performed. The subroutine consists of the FORTRAN statements that follow this statement down to an END, FUNCTION or another SUBROUTINE statement.

The name of the SUBROUTINE does not indicate the mode of any result and hence any letter may be used as its first character. However, the last character of the name must not be F unless it has less than four characters.

Except for the COMMON storage, all variables within a SUBROUTINE (or FUNCTION) are assigned storage that is unique and not used by any other part of the program. Thus the variable X may be used in several SUBROUTINES within a program and each X will be different unless it appears in the same relative position in COMMON statements in each of SUBROUTINES.

No storage is assigned to the dummy variables; on BRLESC, DM will appear in the dictionary instead of an address. If a dummy variable is an array name, then its size must be defined within the subroutine and the size must either be identical with the size of any actual variable used to replace it or else its size must be defined by other dummy variables. Dummy variable array subscripts in DIMENSION statements is a feature of FORTRAN IV and is not allowed in other versions of FORTRAN.

Example: SUBROUTINE POGO(A,XX,L)

2. FUNCTION a(b,c,d,...)

This statement is similar to the SUBROUTINE statement but should be used whenever the subroutine has only one result. No dummy variable should be listed for the result as it is intended that the function will be used in an arithmetic expression and the result is simply used in evaluating the rest of the expression.

The name of the function is a and b,c,d represent nonsubscripted dummy variables. The name of the function does indicate the mode of the result by its first letter and the final character must not be F if there are more than three characters in the name. The first character of the name "a" must be I,J,K,L,M or N if and only if the result is integer.

Within the FUNCTION subprogram, some statement should store a value in a variable that has the same name as the name of the subroutine and this will be used as the result.

There must always be at least one dummy variable for FUNCTION subprograms.

Example: FUNCTION LOW (Q1,T)

3. RETURN

This statement may be used as often as desired within subprograms (SUBROUTINE or FUNCTION) to indicate the point or points at which execution of the subprogram should stop and control should return to the program that is using the subprogram. It should always be used at least once in every subprogram.

4. END

This statement may be used at the end of any subprogram or at the end of the main program. It is not required on BRLESC. All program decks on BRLESC do require the very last card of the entire program deck to be a card that has an E in column 1.

The sense switch options allowed in 709/7090 END statements will be ignored on BRLESC.

Most computers other than BRLESC compile each subprogram as though it is a complete FORTRAN program and only provide a binary card deck that must be assembled with other binary decks to actually run the program. Hence they require an END statement at the end of each main program and each subprogram.

For BRLESC, the main program and all the subprograms must be compiled at the same time and thus can be run without the use of any binary decks. BRLESC has a limit of 30 subprograms used in any one program deck.

XII. PRE-DEFINED FUNCTIONS AND SUBROUTINES

FORTRAN subroutines are separated into two classes, (1) functions are those subroutines that have only one result and hence may be used in arithmetical expressions; and (2) SUBROUTINE subprograms (See Section XI) or other subroutines that may have more than one number as a result and may be used only by CALL statements.

Functions

There are three methods of defining a function. They are

1. Pre-defined functions that may be used by using the pre-defined name.
2. Arithmetic statement functions.
3. FUNCTION subprograms. (See Section XI)

Pre-defined Functions

Unfortunately, there is not much standardization of the pre-defined subroutines that are allowed on various computers. Appendix A lists the pre-defined functions that are allowed on 709/7090 and BRLESC.

Naming Functions

Pre-defined function (and arithmetic statement function) names must always end with F (a total of seven characters are allowed) and must begin with X only if the result is an integer. Variables must never be given a name that is the same as any of the function or subroutine names either with or without the terminal F. For BRLESC, the terminal F is not necessary when using pre-defined functions but is necessary in both the definition and use of arithmetic statement functions.

The naming of FUNCTION subprogram functions uses different rules that are the same as for naming arrays. An initial letter of I,J,K,L,M, or N indicates an integer result and the last character must not be F if there are more than three characters in the name.

Use of Functions

Any of the three types of functions may be used in an arithmetic expression by writing its name in front of a pair of parentheses that enclose the list of arguments. The arguments must correspond in mode and number to the dummy variables used in defining the function. Successive arguments are separated by commas and they may be arithmetic expressions.

For BRLESC, any function may also be used in a CALL statement by including an extra variable name that specifies where to store the result.

Arithmetic Statement Functions

Arithmetic statement functions are functions that can be and are defined by one arithmetic statement at the beginning of a program (or subprogram). The name of the function followed by the dummy arguments enclosed in parentheses are written to the left of the = symbol. The arithmetic expression that describes the function in terms of the dummy variables is written to right of the = symbol. The dummy variables cannot be subscripted. Any variable used in the expression that is not a dummy variable will be identical to the variable of the same name in the program (or subprogram) in which the statement is contained. (An arithmetic statement function definition only applies and can be used only in the program or subprogram in which it is located.)

An arithmetic statement function may use any of the other types of functions and may also use other previously defined arithmetic statement functions. All arithmetic statement functions must precede the first statement that gets executed in the program or subprogram.

The dummy variable names used must indicate the same type of arithmetic that is required when the function is actually used.

Example of defining an arithmetic statement function:

```
FUNF(A,B,C) = A**2 - SINF(B*C)+C
```

Example of using this arithmetic statement function:

```
T = Q + FUNF(X,S + EXPF(V**2),14.)
```

XIII. PRE-DEFINED SUBROUTINES

A subroutine may be pre-defined and exist on the compiler tape or it may be defined by a SUBROUTINE subprogram. (See Section XI.) Subroutines may be given any valid name (no restrictions on the first or last letter) and may only be used by a CALL statement.

The following subroutines are pre-defined in BRLESC FORTRAN:

SETMSI (j)	Set minus sign for input.
SETPSI (j)	Set plus sign for input. (Not necessary, anything not minus is plus.)
SETMSO (j)	Set minus sign for output.
SETPSO (j)	Set plus sign for output.

where j is an integer constant:

0 means blank.

1 means y(12) punch.

2 means x(11) punch.

3 means x or y punch.

SEXAPR(A,B)	Sexadecimal print from A to B.
B.IN	Goes to binary input routine after saving a return jump instruction in 073.
POWER.(A,B,C)	Computes C = A**B where B may be integer of fl. pt.
SINCOS(A,B,C)	Computes B = SINF(A) and C = COSF(A).

Additional pre-defined subroutines will be added in the future.

XIV. FORTRAN PROGRAM CARDS

BRLESC FORTRAN uses the same card format for punching FORTRAN programs that is used by other computers.

Columns:	
1 - 5	Statement number (integer).
6	Continuation Card if not zero or blank.
7 - 72	One FORTRAN statement.
73 - 80	Identification.

The statement number must be a decimal integer. Leading zeros are ignored. Trailing blanks are also ignored, at least on BRLESC.

Column 1 is also used to indicate special types of cards. The following list shows the special characters that indicate special cards:

C	Comment card. Columns 7-80 may be used for comments.
*	709/7090 monitor card or BRLESC control card.
B	Boolean statement card.
D	Double Precision statement card.
I	Complex Arithmetic statement card. (Not allowed on BRLESC.)
F	Used to specify names of subroutines used as arguments.
7-8	End-file signal on 709/7090, ignored on BRLESC.
J	BRLESC only, jump to binary input routine immediately.
E	BRLESC only, is last card of program deck.

Column 6 is used to mark cards that are a continuation of the previous card. It is used as a continuation if Column 6 contains any character other than zero or blank. The only exception to this rule is that the first card of a program may use Column 6 for identification information if it has an * in Column 1. BRLESC does not limit the number of continuation cards allowed but 709/7090 and some other computers restrict a statement to nine continuation cards.

Columns 7-72 contains information, not more than one statement, comments, control information, etc. depending on the type of cards as indicated by Column 1.

Columns 73-80 are ignored by BRLESC and may contain any desired identification.

Blank columns are ignored except when they are in an H field in a FORMAT statement.

Blank cards will be ignored on BRLESC.

Note that it is permissible to use FORAST coding sheets to write FORTRAN programs. It only means that the key punchers must not use Column 6 as part of the statement number and must not allow a statement to go past Column 72. It does not matter whether the statement starts in Column 7 or 11.

<u>F Cards</u>

If the name of a subroutine or function (either pre-defined or defined by a subprogram) is used as an argument for another subroutine or function, its name, without the terminal F, must appear on a card with an F in Column 1. This F card must be in the program (or subprogram) that uses the subroutine as an argument and may be anywhere within that program.

The names of the subroutines are to start in or beyond Column 7 and are separated by commas.

Example: F SIN, EXP, FUN3, ATAN

The terminal F is also to be omitted in the statement that uses the name as an argument. At least on BRLESC, arithmetic statement function names may also be used as arguments for other subroutines without appearing on an F card. However, the terminal F must always be used on arithmetic statement function names on BRLESC.

When writing a subprogram that will accept subroutine arguments, the dummy variable should not have the terminal F in the SUBROUTINE or FUNCTION statement but must have a terminal F added to it when it is used in arithmetic expressions.

XV. BRLESC CONTROL CARDS AND DICTIONARY PRINTING

The use of certain control cards are allowed to affect the compilation of FORTRAN programs. Most of these apply to BRLESC FORTRAN only, although some are also used on 709/7090. All of the control cards are marked with an * in Column 1 with the control information starting in or after Column 7.

* The first card of a program that has an * in Column 1 (except a DATE card) is used as identification and is printed in front of the PUNCH output. Columns 2-80 may be used. (On all other cards with * in Column 1, only Columns 7-72 may be used.)

* CHANGE + AND -

This card reverses the meaning of + and - signs in the program deck, except in FORMAT statements. BRL +(11) and -(12) signs are used initially.

* SETSSW i $\left\{\begin{matrix}\text{UP}\\ \text{DOWN}\end{matrix}\right\}$

This control "statement" allows sense switch i to be "preset" either UP or DOWN. By using this control card, the operator can be relieved of actually setting the sense switches.

* PRTOPU

This control statement causes the compiler to translate all following PRINT statements as though they were PUNCH statements. (Allows card output instead of tape.)

* RTTORC

This control statement causes the compiler to translate all following READ INPUT TAPE or INPUT statements as though they were READ statements. (Use card input instead of tape.)

* WTTOPU

This control statement causes the compiler to translate all following WRITE OUTPUT TAPE or OUTPUT statements as though they were PUNCH statements.

*   LIST
*   SYMBOL TABLE

Either of these causes the storage dictionary to be printed. The asterisk in Column 1 is not required on the LIST card.

The dictionary is printed with names of variables arranged in alphabetical order within each subprogram. Functions (except arithmetic statement functions) and subroutine names will be preceded by two asterisks. Main program names will be preceded only by two blanks and subprogram names will be preceded by one character and one asterisk. The character preceding each subprogram name will be 1,2,...,9,A,B,....T corresponding to the sequence in which the subprograms appeared in the program deck.

Following each name will be the sexadecimal memory address that has been assigned to the name. Following this address, any of the following letters may appear:

A indicates an array name.

X indicates an integer variable.

C indicates the name was in a COMMON statement.

E indicates the name was in an EQUIVALENCE statement.

U indicates the name was used only once.

Statement numbers are printed at the right end of the six character name position and therefore always precede the names of the variables in any program. The compiler usually adds a few names to the dictionary to indicate temporary storage and special subroutines. The name % SUBS. is printed at the end of the dictionary to indicate the length of the predefined subroutines. The subroutines extend from this address down through 0103L and includes all of the input-output storage and subroutines. The % NOS. name printed as the next to last name in the dictionary indicates the length of the "constant pool". This storage, from OSO down to but not including the address printed after % NOS., is used to store the constants and the "array words" required by the program.

For array names, the address assigned in the dictionary is not the initial address of the array, but is the address of the "array word" in the constant pool. The "array word" contains the dimension of the array in its leading two bits, the maximum value of each subscript, (15 bits for each subscript starting at the right end of the B address and going to the left), and the "base address" at the right end of the array word. The "base address" is not the initial address of the array either; for one dimensional arrays, it is one less than the initial address, for two dimensional arrays, it is (Imax. + 1) less than the initial address, and for three dimensional arrays, it is (Imax + Imax · Jmax + 1) less than the initial address. (Imax and Jmax are used here to represent the maximum declared value of the first and second subscripts respectively for an array.)

Whenever the dictionary is printed, the constant pool is also printed so that the programmer can determine the actual storage assigned to arrays. A better method of printing the array storage assignment will be added in the future.

Names in COMMON are assigned last, so the last name in the COMMON assignment within the subprogram that has the most COMMON storage will mark the end of all the storage used by the program. The instructions for the program and all the subprograms are stored first, then all the variables not in COMMON are assigned storage immediately after the instructions and this is followed by those variables in COMMON.

```
*       LIST8
*       LIST (S.CODE)
```

Either of these control cards cause the dictionary and the sexadecimal code for the entire program to be printed. Four instructions are printed on a line with the address of the first one printed at the beginning of the line. The * in Column 1 of LIST (S.CODE) may be omitted unless LIST is the name of an array.

```
*       LIST (B.CODE)
```

This control card causes the entire program and the subroutines it uses to be punched on binary cards with absolute addresses. To use this deck to run the program, it must be preceded by a binary input routine and followed by the standard set of FORTRAN input-output routines and a jump to 073. The * in Column 1 may be omitted unless LIST is the name of an array.

XVI. BRLESC COMPILER ERROR PRINTS

The BRLESC FORTRAN compiler checks for a limited number of types of errors in the program it is compiling. It definitely will not find all possible errors, but some errors will cause one of the error prints listed below. The type of error can be recognized either by the number that follows the word ERROR and precedes the comma or by the "error word" that is printed. The form of the error print is

FORTRAN ERROR t,m <u>Error Word</u> <u>Ident.</u> <u>W</u> <u>First 30 cols. of * Ident.Card</u>

where

t	=	type of error
m	=	ten col. field at which error was detected; m = 0,1,...,7
Error word	=	ten alphanumeric characters that describe the type of error as listed below.
Ident.	=	cols. 73-80 of card at which error was detected.
W	=	rest of the mth field on the card at time of error detection.

TAPE	ERROR WORD	DESCRIPTION
1	ILL.CHAR.	Illegal character on program card.
2	SYM.ST.NO	Symbolic statement number, not all decimal digits.
3	MIXED EXPR	Mixed expression, integer and fl. pt. arithmetic.
4	INT**FLT	Integer raised to fl.pt. power is illegal.
5	IL.RETURN	Illegal RETURN statement, used in main program.
6	NO = IN DO	No equals symbol at proper place in DO statement.
7	SUBPRS. 30	Tried to compile more than 30 subprograms.
8	BIG ADD.ID	Big address is indexed. Program is too big.
9	NO, CP.GOTO	No comma at proper place in computed GOTO statement.
10	ILL.STAT.	Illegal type of statement or too long a name at beginning of arithmetic statement.
11	FLT.INDEX	Subscript involves a fl.pt. number
12	ILL.DIM.	Number of subscripts is not same as dimensionality of the array.

TYPE	ERROR WORD	DESCRIPTION
13	ILL.COMMA	Comma is used improperly in an arithmetic expression.
14	ASD.ST.NO	Assigned statement number; same statement number used twice.
15	COMPLEX AR	Complex arithmetic cards (I in Column 1) not allowed on BRLESC.
16	EQU. TABLE	EQUIVALENCE table is full.
17	COM. ASSND	COMMON name was previously assigned.
18	ARRAY.REF	Array name used before it was defined.
19	DICT.FULL	Dictionary is full.
20	COL.7 NO.	Statement begins with a decimal digit.
21	SENSE > 6	Sense light or Sense Switch number greater than 6.
22	DO NO END	Statement number used in DO never appeared. (It may have been missed due to another error.)
23	LONG NAME	A name was seven or more characters long and seventh character was not an F.
24	IL. EQUALS	Illegal = symbol or arithmetic was specified on the left of the = symbol.
25	IL. - BOOL	Illegal "not" operation on boolean card.
26	IL. / BOOL	Boolean division is undefined.
27	CALL CHAIN	"CHAIN jobs" cannot be done on BRLESC. (Segmentation of program using tape.)
28	IL.**BOOL	Boolean exponentiation is undefined.
29	DRUM STAT.	Drum statements not allowed on BRLESC.
30	IL. IO LIST	Illegal input-output list.
ERROR TAPE 7 FORTRAN		Special error print that usually means machine error.

It must be remembered that the above mentioned cause is only the probable error. Sometimes some type of undetected error later causes one of the detected error prints at a point where no error exists. It also happens that some errors are not detected until the next card has been read. (W=m = 0 when this happens.)

After each error print, the entire card that the compiler currently has in the memory will also be printed. (If w = m = o, the error was probably on the previous card, otherwise it probably is the card that contains the error.)

The ERROR TAPE 7 error print usually indicates a tape error on the temporary storage tape 7. If the right end of the line says PARITY ERROR, then this is indeed a tape error. If the right end of the line has some other characters, they are a symbol that cannot be found in the dictionary and this may be caused by either a tape error or a machine error in the dictionary searching process.

XVII. BRLESC RUN ERROR PRINTS

Some of the pre-defined FORTRAN subroutines used on BRLESC detect certain errors in the data they process. When such an error is detected, a RUN ERROR line is printed and the program is not allowed to continue to run. The error print consists of one line of information of the following form:

<u>RUN ERROR</u> <u>"Error word"</u> <u>Date</u> <u>Cols. 1-30 of Ident. Card</u> <u>LE</u> <u>No</u>.

when "Error word"	is an alphabetic word that identifies the type of error.
Date	is the date.
LE	is the location (in decimal) of the entry to the subroutine that detected the error.
No.	is a number that in some cases was an illegal argument.

Run Error List: (X and Y represent arguments.)

ERROR WORD	SUBROUTINE	REASON	NO.
LOG X NEG	LOGF	$X \leq 0$	X
EXP BIG X	EXPF	$X > 354.89$	$X/\text{Log}_e 2$
ARCSIN 1+	ARCSINF or ARCCOSF	$X > 1+2^{-49}$	$\lvert X \rvert$
SINCOS N S	SINF or COSF or SINCOSF	$\lvert X \rvert /2\pi \geq 16^{13}$	$X/2\pi$
POWER oTO-	POWER. (Exponentiation)	$X = 0$ and $Y \leq 0$	Zero
CVFTOI BIG	XINTF or XFIXF	$\lvert X \rvert \geq 16^{14}$	X

ERROR WORD	SUBROUTINE
END TAPE t	READ TAPE Tried to read beyond information written on tape t.
TAPE TKA u	Persistent tape error on trunk A where u is actual tape switch number and "No." is total number of tape errors.
TAPE TKB u	Same as TAPE TKA u except error is on tape trunk B.
BAD FORMAT	Illegal character in a FORMAT statement.
NO(FORMAT	More right parentheses than left parentheses in a FORMAT statement.
LONG LINE	Output line is more than 170 characters.

XVIII. OPERATION OF THE BRLESC FORTRAN COMPILER

The BRLESC FORTRAN compiler exists on magnetic tape in much the same manner as the FORAST compiler and operates in a very similar manner. Many copies of the compiler and the pre-defined subroutines are on one tape and the tape reading is arranged so that it is checked and automatically corrected by using the next copy on the tape. The tape automatically backs up twenty copies after the last copy on the tape is used. Normally, successive copies are used for compiling successive programs.

Much of the translation is done concurrently with the reading of the program cards (or tape). The partially translated code is put on a temporary tape and the dictionary and constant pool are kept in the memory. After the last card of the program is read (E in Column 1), all unassigned symbols in the dictionary are assigned storage. The memory that will be used by a program is cleared to zeros and then the temporary tape is read, the translation of each instruction is completed and it is stored in the memory for running. Programs are stored from 01040 and may extend to the end of the memory. Next, the subroutines are read from the compiler tape and the ones needed are stored backwards from 09K0. (The standard input-output routines occupy 09K0-103L.)

The efficiency of the generated code is good except for the referencing of arrays with variable subscripts. Such one dimensional referencing causes one extra order to be one, two dimensional referencing causes two extra orders to be done and three dimensional referencing causes four extra orders to be done. These orders are extra in the sense that they would not be needed in the corresponding FORAST or handcoded programs. Subscript expressions and other arithmetic expressions are evaluated as they are written except that instructions involving only constants will be done at compile time. The compiler does not presently make use of the "accumulate" option allowed on BRLESC instructions.

XIX. SPEED OF BRLESC FORTRAN COMPILING

The BRLESC FORTRAN compiler is very fast and hence is designed for "load and go" operation. Programmers are encouraged to keep their FORTRAN programs in symbolic form and translate them each time they are run. This wastes very little if any computer time and is most convenient for the programmer.

Most of the translation is done concurrently with reading the program cards at the present maximum speed of 800 cards per minute. The total time required for translating a program consisting of C cards can be approximated by the formula:

$$\text{time in secs.} = 2 + C/13 + C/75$$

The 2 seconds is compiler tape time, the C/13 is card read time and C/75 allows time for reading the temporary tape and completing the translation. If the program to be translated is put on tape off-line, the C/13 term can be reduced by at least one-half. So the translation rate is about 700 statements per minute from cards or about 1500 statements per minute from tape. (The tape rate will vary considerably with the complexity and length of the statements being translated.)

XX. RUNNING FORTRAN PROGRAMS ON BRLESC

The following list summarizes the steps for compiling and running FORTRAN programs on BRLESC.

1. Have FORTRAN compiler tape on tape switch 15.
2. Have tape switch 7 set to a temporary tape.
3. If have card input, be sure manual read switch 36 is down.
4. If have tape input (program on tape), set manual read switch 36 up and set tape switch 6 to input tape unit.
5. If want all tape output, set manual read switch 38 up and set tape switch 13 to output unit. Also put manual read switch 37 up if this output tape should be rewound at the end of this problem.
6. If programmer specifies any other input or output tapes, mount proper tapes and set proper tape switches. (The programmer may also specify 13 as an output tape without manual read switch 38 being up.)

7. Use "tape start" button to initiate compiling the program.

Halts:

a. 073; program error, initiate only if problems are stacked.

b. N40; end of problem, initiate only if problems are stacked.

c. All other halts or cycles; note PO and NI registers and do a jump to 058. It should soon get to N40.

XXI. MAJOR DIFFERENCES BETWEEN FORAST AND FORTRAN

The following list of some of the basic differences between these two programming languages should be useful to anyone who knows one language and is interested in learning the other one.

1. Statement numbers in FORTRAN must be integer numbers that are used as symbolic names whereas the location field in FORAST may contain any symbolic name and a decimal integer is used as an absolute address.

2. The initial character of a variable name must be alphabetic in FORTRAN and indicates the type of variable. In FORAST, the initial character may be a decimal digit and has no special significance.

3. The type of names used in FORTRAN arithmetic expressions determine the type of arithmetic performed. In FORAST, the type of arithmetic performed is floating point unless changed by a MODE card or by preceding the formula by "FIX" or "INT".

4. The type of a constant is determined in FORTRAN by the presence or absence of a decimal point. In FORAST, constants assume the same type as the type of the statement they are written in.

5. In FORTRAN arithmetic formulas, automatic conversion from one type of variable to another is proveded when the type of the variable on the left of the = symbol is different from the type of the variables used on the right of the = symbol. In FORAST, this conversion must be accomplished by the explicit use of the appropriate subroutine when it is desired.

6. FORAST allows the use of many = symbols to indicate more than one result address in an arithmetic formula while FORTRAN allows only one variable name for a result address.

7. Constant subscripts are enclosed in parentheses in FORTRAN but not in FORAST.

8. All subscripts have an initial value of one in FORTRAN. In FORAST, the initial value may be specified as zero or any positive integer for each array individually.

9. Variable subscripts are allowed in FORTRAN but not in FORAST. FORAST accomplishes the same thing more efficiently by allowing any address to be indexed by a single index register.

10. Three dimensional arrays are allowed in FORTRAN but not in FORAST.

11. FORTRAN allows a multiply and an add or subtract in a subscript expression while FORAST allows only the addition or subtraction of a constant in an indexing expression.

12. FORTRAN allows only one statement per card and FORAST allows more than one. The statement field is columns 7-72 for FORTRAN and columns 11-76 for FORAST.

13. Functions in arithmetic expressions may have more than one argument in FORTRAN but not in FORAST.

14. Implied multiplication is allowed in FORAST but not in FORTRAN (although it does work in some FORTRAN compilers.)

15. The FORTRAN IF statement is very restricted compared to the FORAST IF statement.

16. Absolute addresses are not allowed in FORTRAN but are allowed in FORAST.

XXII. CHECKLIST FOR CONVERTING OTHER COMPUTER FORTRAN PROGRAMS TO BRLESC FORTRAN

1. The first card should be an identification card with an asterisk in Column 1. Columns 2-20 should contain a valid BRLESC problem number.

2. If the signs used in the program are reversed to BRL usage, insert a "CHANGE + AND -" control card after the identification card. (y is - and x is + at BRL.)

3. If input data is included with x minus signs, insert a CALL SETMSI(2) statement where it will be executed before the data is read.

4. Insert an extra card with an E in Column 1 between the program deck and the input data.

5. If the program uses sense switches, it is best to insert control cards to preset them. (* SETSSW i UP or DOWN)

6. If tapes are used, make sure the tape unit numbers used are compatible with BRLESC. (Those over 9 may need to be changed.)

7. Make sure arithmetic statement function names end with F.

8. Make sure that array names of four or more characters do not end with F.

9. Make sure that a nonsubscripted array name has not been used to represent only the first element of the array.

10. Make sure that FORMAT statements do not contain any + signs.

11. Check for proper printer format control characters at the beginning of tape output lines. (A blank character specified puts a 1 single space character on tape.)

12. If desired, change tape output to card output or vice versa by inserting control cards.

13. The program needs to be modified if it contains any of the following: (1) DRUM statements (2) I cards (complex arithmetic), (3) assembly instructions for some other computer or, (4) more memory or tape units than available on BRLESC.

14. Names of variables must not be longer than six characters.

15. If possible, ask the original programmer if they assumed any special characteristics of a particular computer or FORTRAN compiler when writing the program. For example, on some computers, the variable in a DO statement is not always actually used but it is always used on BRLESC.

16. If possible, run a test case that has been run on another computer.

XXIII. SUMMARY OF FORTRAN STATEMENTS

Notations:

s,sl,s2,.....	are statement numbers (look like integer numbers).
i,il,i2,.....	are integer variable names.
m,ml,m2,.....	are integer variable names or integer constants.
ae	represents an arithmetic expression.
b,c,d,e,f	represent any variable names or constants.
t	represents a tape unit number.
f	represents the statement number or variable name of a FORMAT statement.
v,vl,v2	represent variable names.

Specification Statements:

General Form	Brief Description
DIMENSION v,vl,v2,.....	Defines array names and maximum dimensions of each.
EQUIVALENCE (v,vl,..), (v2,v3,..)	Defines synonymous names.
COMMON v,vl,v2,.....	Defines names common between subprograms.
FREQUENCY s(m,ml,..),sl(m2,..)	Provides optimization information. Ignored by BRLESC.

Arithmetic Statements:

v = ae	Evaluates expression ae and stores result in v.
asf(v,vl,...)= ae	Arithmetic statement function where asf represents its name and v,vl,... are the dummy variables.

Control Statements:

GO TO s	DO statement s next.
ASSIGN s TO i	Put <u>address</u> of s into i.
GO TO i, (sl,s2,...)	Do next the statement whose number was last assigned to i by an ASSIGN statement.

General Form	Brief Description
GO TO (s1,s2,...),i	Do statement si next.
DO s i = m1,m2,m3	Repeat statements to and including s with i = m1,m1 + m3, m1 + 2m3,... until i > m2.
DO s i = m1,m2	Same as above with m3 = 1.
IF(ae)s1,s2,s3	Do statement s1 next if ae is negative; s2 next if ae is zero and s3 next if ae is positive.
CONTINUE	Dummy statement.
STOP or STOP w	End of execution of main program. (w is octal no.)
PAUSE or PAUSE w	Computer halts. (Displays octal no. w.)
CALL name (v,v1,v2,...)	Perform the subroutine specified by "name".
IF(SENSE SWITCH r)s1,s2	Do statement s1 next if switch r is down, do s2 next if it is up.
SENSE LIGHT r	For r = 0 turn all sense lights off. For r = 1,2,3, or 4, turn light r on.
IF(SENSE LIGHT r)s1,s2	Do statement s1 or s2 next if sense light r is on or off respectively.

IF ACCUMULATOR OVERFLOW s1,s2
IF QUOTIENT OVERFLOW s1,s2
IF DIVIDE CHECK s1,s2
	These are special statements to check certain overflow indicators. Statement s1 or s2 is done next if indicator is on or off respectively.

Subprogram Statements:

SUBROUTINE name (v,v1,v2,...)	Defines the name and beginning of a subroutine. v,v1,v2,... are the dummy variables.
FUNCTION name (v,v1,v2,...)	Defines the name and beginning of a function subprogram.
RETURN	Indicates an execution exit of a subprogram.
END	Marks the end of a subprogram.

General Form	Brief Description
Input-Output Statements:	
FORMAT (Special Specifications)	Describes the fields for input-output data.
READ f, list	Read cards.
PUNCH f, list	Punch cards.
PRINT f, list	Print data, (on-line on some computers, off line on BRLESC).
READ INPUT TAPE t,f, list	Read alphanumeric tape.
WRITE OUTPUT TAPE t,f,list	Write alphanumeric tape.
READ TAPE t, list	Read binary tape.
WRITE TAPE t, list	Write binary tape.
END FILE t	Write end-of-file mark on tape.
BACKSPACE t	Move tape back one record.
REWIND t	Rewind tape.
READ DRUM m,ml,list	Read drum. (Illegal on BRLESC.)
WRITE DRUM m,ml,list	Write drum. (Illegal on BRLESC.)

ACKNOWLEDGEMENTS

Mr. Alfred Anderson reviewed the text and programmed the subroutine that reads and prints decimal numbers in the object programs. Mr. Michael Romanelli also reviewed the text.

Lloyd W Campbell

LLOYD W. CAMPBELL

Glenn Beck

GLENN BECK

REFERENCES

1. Campbell, L. and Beck, G. The Instruction Code for the BRL Electronic Scientific Computer (BRLESC), Ballistic Research Laboratories Memorandum Report No. 1379, November 1961.

2. Campbell, L. and Beck, G. The FORAST Programming Language for ORDVAC and BRLESC, Ballistic Research Laboratories Report No. 1172, August 1962.

3. IBM Reference Manual, 709/7090 FORTRAN Programming System (Form C28-6054-2), 1961.

4. IBM General Information Manual, FORTRAN (Form F28-8074-1), 1961.

5. IBM General Information Manual, Programmer's Primer for FORTRAN Automatic Coding System for the IBM 704 Data Processing System (Form F28-6019), 1957.

6. IBM Reference Manual, FORTRAN Automatic Coding System for the IBM 704 Data Processing System (Form C28-6003), 1958.

7. IBM Reference Manual, 709/7090 FORTRAN Operations (Form C28-6066-3), 1961.

APPENDIX A

LIST OF PRE-DEFINED FUNCTIONS FOR 709/7090 AND BRLESC
(F indicates fl.pt. and I indicates integer)

NAME	ARGUMENT	RESULT	NO. of ARGS.	DEFINITION
ABSF	F	F	1	Absolute value.
XABSF	I	I	1	Absolute value.
INTF	F	F	1	Truncation to whole number.
XINTF	F	I	1	Convert fl.pt. no. to integer.
MODF	F	F	2	Arg.1(mod Arg.2).
XMODF	I	I	2	Arg.1(mod Arg.2).
MAXOF	I	F	≥ 2	Chooses largest argument.
MAX1F	F	F	≥ 2	Chooses largest argument.
XMAXOF	I	I	≥ 2	Chooses largest argument.
XMAX1F	F	I	≥ 2	Chooses largest agrument.
MINOF	I	F	≥ 2	Chooses smallest argument.
MIN1F	F	F	≥ 2	Chooses smallest argument.
XMINOF	I	I	≥ 2	Chooses smallest argument.
XMIN1F	F	I	≥ 2	Chooses smallest argument.
FLOATF	I	F	1	Convert integer to fl. pt.
XFIXF	F	I	1	Convert fl.pt. to integer.
SIGNF	F	F	2	Transfer sign of Arg.2 to Arg.1.
XSIGNF	I	I	2	Transfer sign of Arg.2 to Arg.1.
DIMF	F	F	2	Arg.1 - minimum (Arg.1, Arg.2).
XDIMF	I	I	2	Arg.1 - minimum (Arg.1, Arg.2).
XLOCF	F or I	I	1	Stores the address of the Arg.
SQRTF	F	F	1	Square root.
SINF	F	F	1	Sine (argument in radians)
COSF	F	F	1	Cosine (argument in radians)
LOGF	F	F	1	Natural logarithm.
EXPF	F	F	1	Exponential.
ATANF	F	F	1	Arctangent (result in radians)
TANHF	F	F	1	Hyperbolic tangent.
For BRLESC only:				
ARCSINF	F	F	1	Arcsine.
ARCCOSF	F	F	1	Arcosine.
ARCTANF	F	F	1	Arctangent (same as ATANF).

APPENDIX B. THREE EXAMPLES OF FORTRAN PROGRAMS

(PROGRAM,INPUT DATA, AND OUTPUT ARE LISTED)

```
*     EXAMPLE 1 MULTIPLY TWO VECTORS A*B   L.W. CAMPBELL
      DIMENSION A(10),B(10),C(10)
      READ 2,A,B
      DO 3 I=1,10
    3 C(I)=A(I)*B(I)
      PUNCH 4,C
      STOP
    4 FORMAT(14HVECTOR PRODUCT/(5E14.7))
      END
END   (THIS CARD REQUIRED ONLY ON BRLESC.)
          14.1          60.35          22.8          91.7         374.18
          36.2         193.44         83.61         2.648            9.8
          4.21           8.23          15.9          7.77           88.1
           2.7            3.0        8.1118          19.1          42.44
```

```
OCT.25,63   BRLESC   FORTRAN 2
*     EXAMPLE 1 MULTIPLY TWO VECTORS A*B   L.W. CAMPBELL

VECTOR PRODUCT
 0.5936100E 02 0.4966805E 03 0.3625200E 03 0.7125090E 03 0.3296526E 05
 0.9774000E 02 0.5803200E 03 0.6782276E 03 0.5057680E 02 0.4159120E 03
```

```
*     EXAMPLE 2 FIND SMALLEST NUMBER IN ARRAY F, L.CAMPBELL
      DIMENSION F(20)
      READ 2,F
      SMALL=F(1)
      DO 9 J=2,20
      IF(SMALL-F(J))9,9,8
    8 SMALL=F(J)
    9 CONTINUE
      PUNCH 3,SMALL
      STOP
    3 FORMAT(11HSMALLEST F=F13.6)
      END
END
          14.1          60.35           22.8           91.7         374.18
          36.2         193.44          83.61          2.648            9.8
          4.21           8.23           15.9           7.77           88.1
           2.7            3.0         8.1118           19.1          42.44


OCT.25,63   BRLESC   FORTRAN 2
*     EXAMPLE 2 FIND SMALLEST NUMBER IN ARRAY F, L.CAMPBELL


SMALLEST F=     2.648000
```

```
*      EXAMPLE 22 FROM BRL REPORT 1209 CODED IN FORTRAN-L.CAMPBELL
C      USE BISECTION METHOD TO FIND ROOT OF F(X)=X**3-X-1  IN INTERVAL (1,2)
    11 FORMAT(5X,1HX10X4HF(X)//)
    21 FORMAT(1P2E15.7)
    31 FORMAT(24HCONDITIONS NOT SATISFIED)
       X=1.
       X1=2.
       EPS=.00001
       ASSIGN 1 TO K
       PUNCH 11
    44 F=X*(X*X-1.)-1.
       PUNCH21,X,F
       GOTOK,(1,4,7)
     1 F0=F
       IF(F)2,15,15
    15 XP=X
       GOTO 3
     2 XN=X
     3 X=X1
       ASSIGN 4 TO K
       GOTO 44
     4 F1=F
       IF(F)5,45,45
    45 XP=X
       GOTO 6
     5 XN=X
     6 ASSIGN 7 TO K
       IF(F0*F1)66,65,65
    65 PUNCH 31
    67 STOP
    66 X=(XN+XP)/2.
       GOTO 44
     7 IF(ABSF(F)-EPS)67,71,71
    71 IF(F)8,72,72
    72 XP=X
       GOTO 66
     8 XN=X
       GOTO 66
       END
END


OCT.25,63   BRLESC  FORTRAN 2
*      EXAMPLE 22 FROM 8RL REPORT 1209 CODED IN FORTRAN-L.CAMPBELL

     X            F(X)

  1.0000000E 00 -1.0000000E 00
  2.0000000E 00  5.0000000E 00
  1.5000000E 00  8.7500000E-01
  1.2500000E 00 -2.9687500E-01
  1.3750000E 00  2.2460938E-01
  1.3125000E 00 -5.1513672E-02
  1.3437500E 00  8.2611084E-02
  1.3281250E 00  1.4575958E-02
  1.3203125E 00 -1.8710613E-02
  1.3242187E 00 -2.1279454E-03
  1.3261719E 00  6.2088296E-03
  1.3251953E 00  2.0366507E-03
  1.3247070E 00 -4.6594883E-05
  1.3249512E 00  9.9479097E-04
  1.3248291E 00  4.7403882E-04
  1.3247681E 00  2.1370716E-04
  1.3247375E 00  8.3552438E-05
  1.3247223E 00  1.8477852E-05
  1.3247147E 00 -1.4058747E-05
```

DISTRIBUTION LIST

No. of Copies	Organization
20	Commander Defense Documentation Center ATTN: TIPCR Cameron Station Alexandria, Virginia 22314
1	Director Advanced Research Projects Agency Department of Defense Washington, D. C. 20301
1	Director, National Security Agency ATTN: R/D 36, Chief Engineering Research Division Fort George G. Meade, Maryland 20755
1	Commanding General U.S. Army Materiel Command ATTN: AMCRD-RP-B Washington, D. C. 20315
1	Commanding Officer Frankford Arsenal Philadelphia, Pennsylvania 19137
1	Commanding Officer Picatinny Arsenal ATTN: Feltman Research and Engineering Laboratories Dover, New Jersey 07801
1	Commanding General U.S. Army Missile Command Redstone Arsenal, Alabama 35809
1	Commanding Officer Watertown Arsenal Watertown, Massachusetts 02172
1	Commanding General U.S. Army Munitions Command Dover, New Jersey 07801

No. of Copies	Organization
1	Commanding General U.S. Army Tank - Automotive Center Land Locomotion Research Laboratory Warren, Michigan 48090
1	Commanding General Army Weapons Command Rock Island, Illinois 61200
1	Commanding General White Sands Missile Range New Mexico 88002
1	Director Major Item Supply Management Agency Letterkenny Army Depot Chambersburg, Pennsylvania 17201
1	Commanding General U.S. Army Chemical Corps Research & Development Command Washington 25, D. C.
1	Commanding General U.S. Army Chemical Research & Development ATTN: Dr. Carl M. Herget Edgewood Arsenal, Maryland 21040
1	Commanding Officer U.S. Army CBR Combat Developments Agency ATTN: David T. Shepard, Director Data Processing Center Edgewood Arsenal, Maryland 21040
1	Commanding General U.S. Army Chemical Center Corps Engineering Command Edgewood Arsenal, Maryland 21040
1	Director U.S. Army Nuclear Defense Laboratory Edgewood Arsenal, Maryland 21040

DISTRIBUTION LIST

No. of Copies	Organization
1	Commanding Officer Operations Research Group Edgewood Arsenal, Maryland 21040
1	Director U.S. Army Chemical Corps Quality Assurance Technical Agency Edgewood Arsenal, Maryland 21040
1	Chairman U.S. Army Chemical Corps Technical Committee Edgewood Arsenal, Maryland 21040
1	Commanding General U.S. Army Chemical Corps Proving Ground Dugway Proving Ground Dugway, Utah 84022
1	Director U.S. Army Biological Laboratories Fort Detrick, Maryland 21701
1	Commanding Officer U.S. Army Corps of Engineers Army Reactors Group Fort Belvoir, Virginia 22060
1	Director U.S. Army Corps of Engineers Geodessy Intelligence and Mapping Research & Development Agency Fort Belvoir, Virginia 22060
1	Commanding Officer U.S. Army Corps of Engineers Polar Research & Development Center Fort Belvoir, Virginia 22060

No. of Copies	Organization
1	Commanding General U.S. Army Research & Development Laboratories Fort Belvoir, Virginia 22060
1	Commanding Officer U.S. Army Corps of Engineers Army Reactors Group Germantown, Maryland
1	Commanding Officer U.S. Army Corps of Engineers Waterways Experiment Station P.O. Box 631 Vicksburg, Mississippi
1	U.S. Army Corps of Engineers Special Assistant for Nuclear Power Building T-7 Washington 25, D. C.
1	Director U.S. Army Corps of Engineers Cold Regions Research and Engineering Laboratory 1215 Washington Avenue Wilmette, Illinois
1	Director U.S. Army Medical Research and Nutrition Laboratory Denver, Colorado
1	Commanding Officer U.S. Army Medical Research and Development Command Washington 25, D. C.
1	Commanding Officer U.S.Army Medical Unit Fort Detrick, Maryland
1	Commanding Officer U.S. Army Medical Research Laboratory Fort Knox, Kentucky 40120

DISTRIBUTION LIST

No. of Copies	Organization
1	Commanding Officer; U.S. Army Signal Engineering Agency; Arlington Hall Station; Arlington, Virginia
1	Commanding General; U.S. Army Signal Missile Support Agency; White Sands Missile Range; New Mexico 88002
1	Commanding Officer; U.S. Army Signal Intelligence Agency; Arlington Hall Station; Arlington, Virginia
1	Commanding Officer; U.S. Army Signal Electronic Research Unit; P.O. Box 205; Mountain View, California
1	Commanding Officer; U.S. Army Signal Avionics Field Office; P.O. Box 209; St. Louis 66, Missouri
1	Chief Signal Officer; Department of the Army; Washington 25, D. C.
1	Commandant; U.S. Army Signal Corps School; ATTN: Officer Department; Fort Monmouth, New Jersey 07703
1	Commanding General; U.S. Army Electronic Proving Ground; Fort Huachuca, Arizona 85613
1	Commanding General; U.S. Army Electronics Research and Development Laboratory; ATTN: Data Equipment Branch; Fort Monmouth, New Jersey 07703
1	Commanding General; U.S. Army Electronics Command; ATTN: ANSEL-CB; Fort Monmouth, New Jersey 07703
1	Director; U.S. Army Quartermaster Research and Engineering Field Evaluation Agency; Fort Lee, Virginia 23801
1	Commanding Officer; U.S. Army Transportation Materiel Command; 12th and Spruce Streets; St. Louis, Missouri
1	Commanding General; U.S. Army Transportation Research Command; Fort Eustis, Virginia 23604
1	Commanding General; U.S. Continental Army Command; Fort Monroe, Virginia 23351
1	Commanding General; U.S. Army Combat Developments Command; ATTN: CDCRE-C; Fort Belvoir, Virginia 22060
1	Commanding General; USACDC Combined Arms Group; Fort Leavenworth, Kansas 66027
1	Commanding General; U.S. Army Combat Developments Command; ATTN: CCISG; Fort Belvoir, Virginia 22060

DISTRIBUTION LIST

No. of Copies	Organization
1	Commandant U.S. Army Artillery & Guided Missile School Fort Sill, Oklahoma 73503
1	Commandant U.S. Army Guided Missile School Redstone Arsenal, Alabama 35809
1	Army Research Office 3045 Columbia Pike Arlington, Virginia
1	Commanding Officer Army Research Officer (Durham) Box CM, Duke Station Durham, North Carolina 27706
1	Commandant Command & General Staff College ATTN: Computing Facility Fort Leavenworth, Kansas 66027
1	Commanding General Fort George G. Meade ATTN: Computing Facility Maryland 20755
1	Commanding Officer U.S. Army Communications Agency The Pentagon Washington 25, D. C.
1	Professor of Ordnance U.S. Military Academy West Point, New York 10996
3	Chief, Bureau of Naval Weapons ATTN: DLI-3 Department of the Navy Washington, D. C. 20360
1	Commander U.S. Naval Ordnance Laboratory White Oak Silver Spring 19, Maryland

No. of Copies	Organization
1	Commanding Officer U.S. Naval Ordnance Laboratory Corona, California 91720
1	Commander U.S. Naval Ordnance Test Station China Lake, California 93557
1	Library U.S. Naval Postgraduate School ATTN: Technical Reports Section Monterey, California
1	Director U.S. Naval Research Laboratory ATTN: Code 492 Washington, D. C. 20390
1	Commander U.S. Naval Weapons Laboratory ATTN: Computation & Analysis Branch Dahlgren, Virginia 22448
1	Chief, Bureau of Ships ATTN: Computing Facility Department of the Navy Washington, D. C. 20360
1	Chief, Bureau of Yards & Docks ATTN: Data Processing and Analysis Branch Department of the Navy Washington, D. C. 20360
1	Chief of Naval Operations Department of the Navy Washington, D. C. 20360
1	Commanding Officer U.S. Naval Air Development Center Johnsville, Pennsylvania
1	Commanding Officer U.S. Naval Air Test Center ATTN: Armament Test U.S. Naval Air Station Patuxent River, Maryland

DISTRIBUTION LIST

No. of Copies	Organization
2	Commander U.S. Naval Missile Center ATTN: Simulation Branch Systems Department Range Operations Department Code 3280 Point Mugu, California 93041
1	Commander Naval Engineering Experiment Station ATTN: Applied Math Office Code 502 Annapolis, Maryland
1	Commanding Officer and Director David W. Taylor Model Basin ATTN: Technical Library Code 042 Washington, D. C. 20007
1	Commanding Officer & Director U.S. Naval Radiological Defense Laboratory San Francisco 24, California
1	Director U.S. Naval Supersonic Laboratory Massachusetts Institute of Technology ATTN: Computer Facility 560 Memorial Drive Cambridge, Massachusetts
1	Superintendent U.S. Naval Academy ATTN: Weapons Department Annapolis, Maryland
1	Commanding Officer Fleet Operations Control Center US Pacific Fleet ATTN: F. N. Quinn Navy No. 509 Fleet Post Office San Francisco, California
1	Commandant US Marine Corps Code AX Washington, 25, D. C.
1	Director Marine Corps Landing Force Development Center Marine Corps Schools Quantico, Virginia 22134
1	AEDC Arnold Air Force Station Tennessee 37389
1	AFFTC (FTFSE) Edwards Air Force Base California 93523
1	AFMTC Patrick Air Force Base Florida
1	AFMDC Holloman Air Force Base New Mexico 88330
1	AFCRL L. G. Hanscom Field Bedford, Massachusetts 01731
1	TAC (DCRS) Langley Air Force Base Virginia 23365

DISTRIBUTION LIST

No. of Copies	Organization
1	AUL (3T-AUL-60-118) Maxwell Air Force Base Alabama 36112
1	AFIT (NCLI) Wright-Patterson Air Force Base Ohio 45433
1	ASD (ASNCD) Wright-Patterson Air Force Base Ohio 45433
1	AFWL Kirtland Air Force Base New Mexico
1	Headquarters, USAF (AFAAC) Washington, D. C. 20330
1	Headquarters, USAF (AFADA) Washington, D. C. 20330
1	Headquarters, USAF (AFNIN3) Washington, D. C. 20330
1	USAFA United States Air Force Academy Colorado 80840
1	Central Intelligence Agency OCR/Library/ILS ATTN: Code 163 Washington, D. C. 20505
1	Director Air Weather Service Climatic Center 225 D Street, S.E. Washington 25, D. C.

No. of Copies	Organization
1	Federal Aviation Agency National Aviation Facilities Experimental Station ATTN: Simulation & Computation Branch Atlantic City, New Jersey
1	Federal Aviation Agency ATTN: Data Processing Branch- Aircraft Management Division Bureau of Flight Standards P.O. Box 1082 Oklahoma City, Oklahoma
1	Director National Aeronautics and Space Administration ATTN: Mr. R.E. Liettell 1520 H Street, N.W Washington, D. C. 20546
1	Director National Aeronautics and Space Administration Flight Research Center ATTN: Computer Facility Box 273 Edwards, California
2	Director National Aeronautics and Space Administration Goddard Space Flight Center ATTN: Tracking & Data Systems I. Mortimer Datz- Computer Operations Branch Data Systems Division Anacostia Naval Station Washington 25, D. C.

DISTRIBUTION LIST

No. of Copies

1 Director
National Aeronautics and Space Administration
Lewis Research Center
ATTN: Computer Faciltiy
Cleveland Airport
Cleveland, Ohio

3 Director
National Bureau of Standards
ATTN: Mr. Paul Neissner
Components & Techniques Section-Data Processing Systems Division
Dr. S. N. Alexander
Computation Laboratory
232 Dynamomenter Building
Washington 25, D. C.

1 U. S. Department of Commerce
Bureau of Census
ATTN: Computer Facility
Federal Office Building No. 3
Suitland, Maryland

1 Brookhaven National Laboratory
ATTN: Computer Facility
Upton, New York

1 Oak Ridge National Laboratory
P. O. Box P
Oak Ridge, Tennessee

1 Research Analysis Corporation
ATTN: Computer Facility
6935 Arlington Road
Bethesda, Maryland
Washington, D. C. 20014

1 The Johns Hopkins University
Applied Physics Laboratory
ATTN: Computer Facility
8621 Georgia Avenue
Silver Spring, Maryland

No. of Copies — Organization

1 California Institute of Technology
Jet Propulsion Laboratory
ATTN: Computer Facility
4800 Oak Grove Drive
Pasadena, California 91103

1 Ampex Computer Products Company
9937 Jefferson Boulevard
Culver City, California

1 Datatrol Corporation
Consulting & Programming Services
ATTN: Mr. Cooper, Vice President
8113A Fenton Street
Silver Spring, Maryland

1 E. I. DuPont DeNemours Company
Engineering Department
ATTN: Theodore Baumeister, III
Wilmington 98, Delaware

1 Engineering Research Associates
Division of Remington Rand, Inc.
1902 W. Minnehaha Avenue
St. Paul, Minnesota

1 General Mills
Electronics Group
1000 16th Street, N. W.
506 Solar Building
Washington 6, D. C.

1 International Business Machine Corporation
Engineering Laboratory
ATTN: Customer Executive Education Department
San Jose, California

1 Raytheon Manufacturing Company
P. O. Box 398
Bedford, Massachusetts

DISTRIBUTION LIST

No. of Copies	Organization
1	Remington Rand Univac Division of Sperry Rand Corporation ATTN: Systems Analysis 1900 W. Allegheny Avenue St. Paul, Minnesota
1	Science Research Association Incorporated 259 East Erie Street ATTN: Mr. Don Shepherd Project Director Chicago 11, Illinois
1	Technitrol Engineering Corporation 1952 E. Alleghany Avenue Philadelphia 34, Pennsylvania
1	California Institute of Technology ATTN: Comptroller Pasadena, California
1	Columbia University Electronics Research Laboratories 632 West 125 Street New York 27, New York
1	Columbia University Lewis Cyclation Laboratory ATTN: Computer Facility Box 137 Irvington on Hudson, New York
1	Cornell University ATTN: Coordinator of Research Ithaca, New York
1	Dartmouth College ATTN: Computation Center Hanover, New Hampshire
1	The George Washington University ATTN: Logistics Research Project 707 22nd Street, N.W. Washington 7, D. C.
1	Georgia Institute of Technology Engineering Experiment Station ATTN: Rich Electronic Computer Center Atlanta 13, Georgia
1	Harvard University Computation Laboratory Cambridge 38, Massachusetts
1	Indiana University ATTN: Research Computing Center Bloomington, Indiana
1	Iowa State University of Science and Technology Engineering Experiment Station ATTN: Cyclone Computer Laboratory Ames, Iowa
1	The Johns Hopkins University ATTN: Computation Center 34th and Charles Streets Baltimore 18, Maryland
1	Lehigh University ATTN: Computer Facility Bethehem, Pennsylvania
1	Marquette University ATTN: Computing Center 1515 West Wisconsin Avenue Milwaukee, Wisconsin
1	Michigan State University College of Engineering ATTN: Computer Laboratory East Lansing, Michigan

DISTRIBUTION LIST

No. of Copies — Organization

1 Missouri School of Mines
and Metallurgy
ATTN: Computer Facility
Rolla, Missouri

1 New York University
College of Engineering
ATTN: Computation &
Statistical Laboratory
University Heights
New York 53, New York

1 Oklahoma State University
The Computing Center
ATTN: Department of
Mathematics
Stillwater, Oklahoma

1 Oregon State College
Department of Mathematics
ATTN: W. E. Milne
Corvallis, Oregon

1 Polytechnic Institute of
Brooklyn
ATTN: Mr. Warren Roes
333 Jay Street
Brooklyn 1, New York

1 Princeton University
Mathematics Department
Princeton, New Jersey

1 Stanford University
ATTN: Computation Center
Stanford, California 13201

1 University of Alberta
Department of Mathematics
ATTN: Professor John McNamee
Edmonton, Alberta, Canada

No. of Copies — Organization

1 University of California
942 Hilldale Avenue
Berkeley, California

1 University of Delaware
Newark, Delaware

1 University of Illinois
Department of Mathematics
Urbana, Illinois

1 University of Pennsylvania
The Moore School of Electrical
Engineering
ATTN: Mr. Ingerman
Philadelphia 4, Pennsylvania

1 Professor Bruce Charters
Computing Laboratory
Brown University
Providence, Rhode Island

1 Dr. L. H. Thomas
Watson Scientific Computing
Laboratory
612 W. 116th Street
New York 27, New York

1 Mary Broadhead
12 Westfall Avenue
Troy, New York

4 Australian Group
c/o Military Attache
Australian Embassy
2001 Connecticut Avenue, N. W.
Washington, D. C. 20008

DISTRIBUTION LIST

No. of Copies	Organization
10	The Scientific Information Officer Defence Research Staff British Embassy 3100 Massachusetts Avenue, N.W. Washington, D. C. 20008
4	Defence Research Member Canadian Joint Staff 2450 Massachusetts Avenue, N.W. Washington, D. C. 20008

Aberdeen Proving Ground

Chief, TIB

Air Force Liaison Office
Marine Corps Liaison Office
Navy Liaison Office
CDC Liaison Office

D & PS Branch Library

AD ____ Accession No. ____ UNCLASSIFIED

Ballistic Research Laboratories, APG
BRLESC FORTRAN
Lloyd W. Campbell and Glenn Beck

BRL Report No. 1229 November 1963

RDT & E Project No. 1M010501A003
UNCLASSIFIED Report

Mathematical computers - coding
Mathematical computers - operation

FORTRAN is a popular programming language that has been implemented on many computers. It is now available on Ballistic Research Laboratories' BRLESC computer. This report describes the FORTRAN language in general and includes specific details about its implementation on BRLESC.

AD ____ Accession No. ____ UNCLASSIFIED

Ballistic Research Laboratories, APG
BRLESC FORTRAN
Lloyd W. Campbell and Glenn Beck

BRL Report No. 1229 November 1963

RDT & E Project No. 1M010501A003
UNCLASSIFIED Report

Mathematical computers - coding
Mathematical computers - operation

FORTRAN is a popular programming language that has been implemented on many computers. It is now available on Ballistic Research Laboratories' BRLESC computer. This report describes the FORTRAN language in general and includes specific details about its implementation on BRLESC.

AD ____ Accession No. ____ UNCLASSIFIED

Ballistic Research Laboratories, APG
BRLESC FORTRAN
Lloyd W. Campbell and Glenn Beck

BRL Report No. 1229 November 1963

RDT & E Project No. 1M010501A003
UNCLASSIFIED Report

Mathematical computers - coding
Mathematical computers - operation

FORTRAN is a popular programming language that has been implemented on many computers. It is now available on Ballistic Research Laboratories' BRLESC computer. This report describes the FORTRAN language in general and includes specific details about its implementation on BRLESC.

AD ____ Accession No. ____ UNCLASSIFIED

Ballistic Research Laboratories, APG
BRLESC FORTRAN
Lloyd W. Campbell and Glenn Beck

BRL Report No. 1229 November 1963

RDT & E Project No. 1M010501A003
UNCLASSIFIED Report

Mathematical computers - coding
Mathematical computers - operation

FORTRAN is a popular programming language that has been implemented on many computers. It is now available on Ballistic Research Laboratories' BRLESC computer. This report describes the FORTRAN language in general and includes specific details about its implementation on BRLESC.

AD____________________ Accession No.____________________ UNCLASSIFIED

Ballistic Research Laboratories, APG
BRLESC FORTRAN
Lloyd W. Campbell and Glenn Beck

BRL Report No. 1229 November 1963

RDT & E Project No. 1M010501A003
UNCLASSIFIED Report

Mathematical computers - coding
Mathematical computers - operation

FORTRAN is a popular programming language that has been implemented on many computers. It is now available on Ballistic Research Laboratories' BRLESC computer. This report describes the FORTRAN language in general and includes specific details about its implementation on BRLESC.

AD____________________ Accession No.____________________ UNCLASSIFIED

Ballistic Research Laboratories, APG
BRLESC FORTRAN
Lloyd W. Campbell and Glenn Beck

BRL Report No. 1229 November 1963

RDT & E Project No. 1M010501A003
UNCLASSIFIED Report

Mathematical computers - coding
Mathematical computers - operation

FORTRAN is a popular programming language that has been implemented on many computers. It is now available on Ballistic Research Laboratories' BRLESC computer. This report describes the FORTRAN language in general and includes specific details about its implementation on BRLESC.

AD____________________ Accession No.____________________ UNCLASSIFIED

Ballistic Research Laboratories, APG
BRLESC FORTRAN
Lloyd W. Campbell and Glenn Beck

BRL Report No. 1229 November 1963

RDT & E Project No. 1M010501A003
UNCLASSIFIED Report

Mathematical computers - coding
Mathematical computers - operation

FORTRAN is a popular programming language that has been implemented on many computers. It is now available on Ballistic Research Laboratories' BRLESC computer. This report describes the FORTRAN language in general and includes specific details about its implementation on BRLESC.

AD____________________ Accession No.____________________ UNCLASSIFIED

Ballistic Research Laboratories, APG
BRLESC FORTRAN
Lloyd W. Campbell and Glenn Beck

BRL Report No. 1229 November 1963

RDT & E Project No. 1M010501A003
UNCLASSIFIED Report

Mathematical computers - coding
Mathematical computers - operation

FORTRAN is a popular programming language that has been implemented on many computers. It is now available on Ballistic Research Laboratories' BRLESC computer. This report describes the FORTRAN language in general and includes specific details about its implementation on BRLESC.